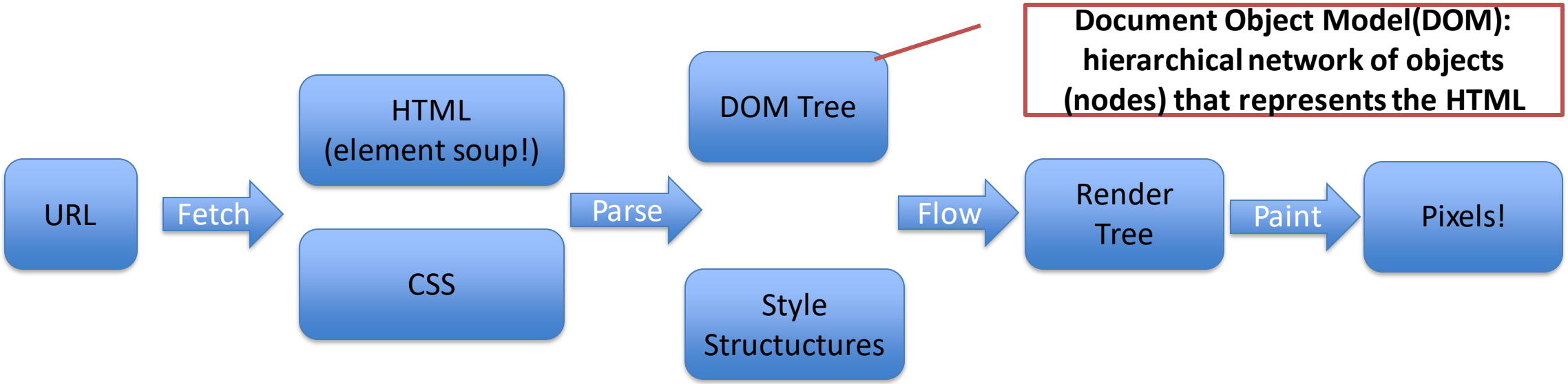
The Web Browser

An event-driven environment

Rendering Process



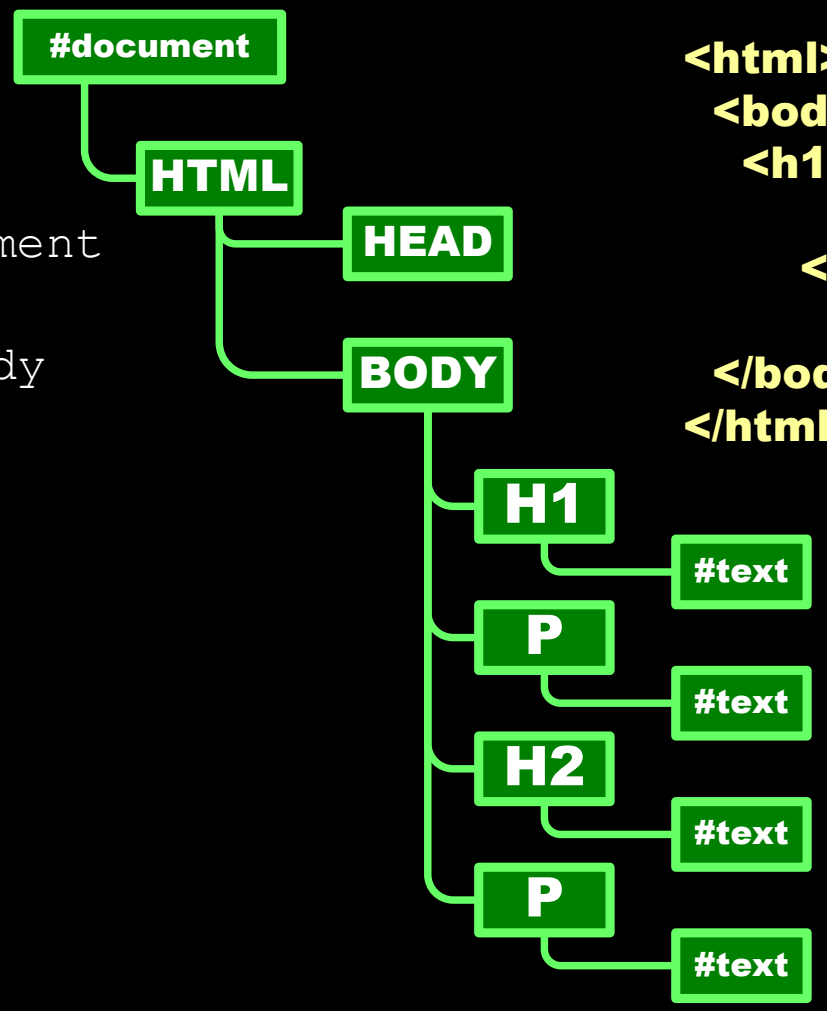
Repaints/Reflows

- Changes to the input info used to construct the rendering tree will cause all/part of the screen to be rendered
  - E.g. changing the Document Object Model.
- Usually resulting from user actions
  - E.g. click button/resize/drag & drop

Button

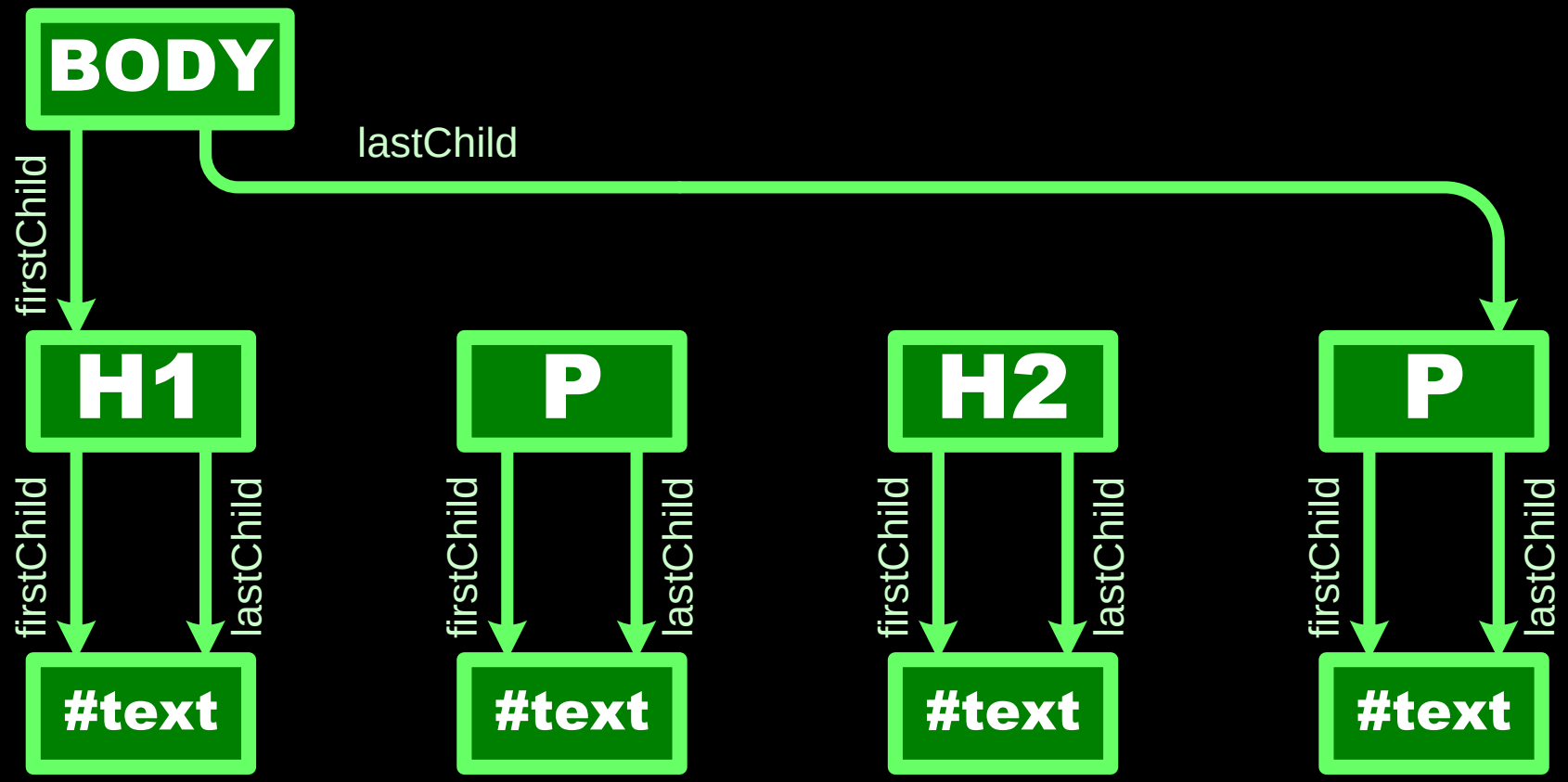
Document Tree Structure (aka DOM)

```
document

document.
documentElement

document.body
```

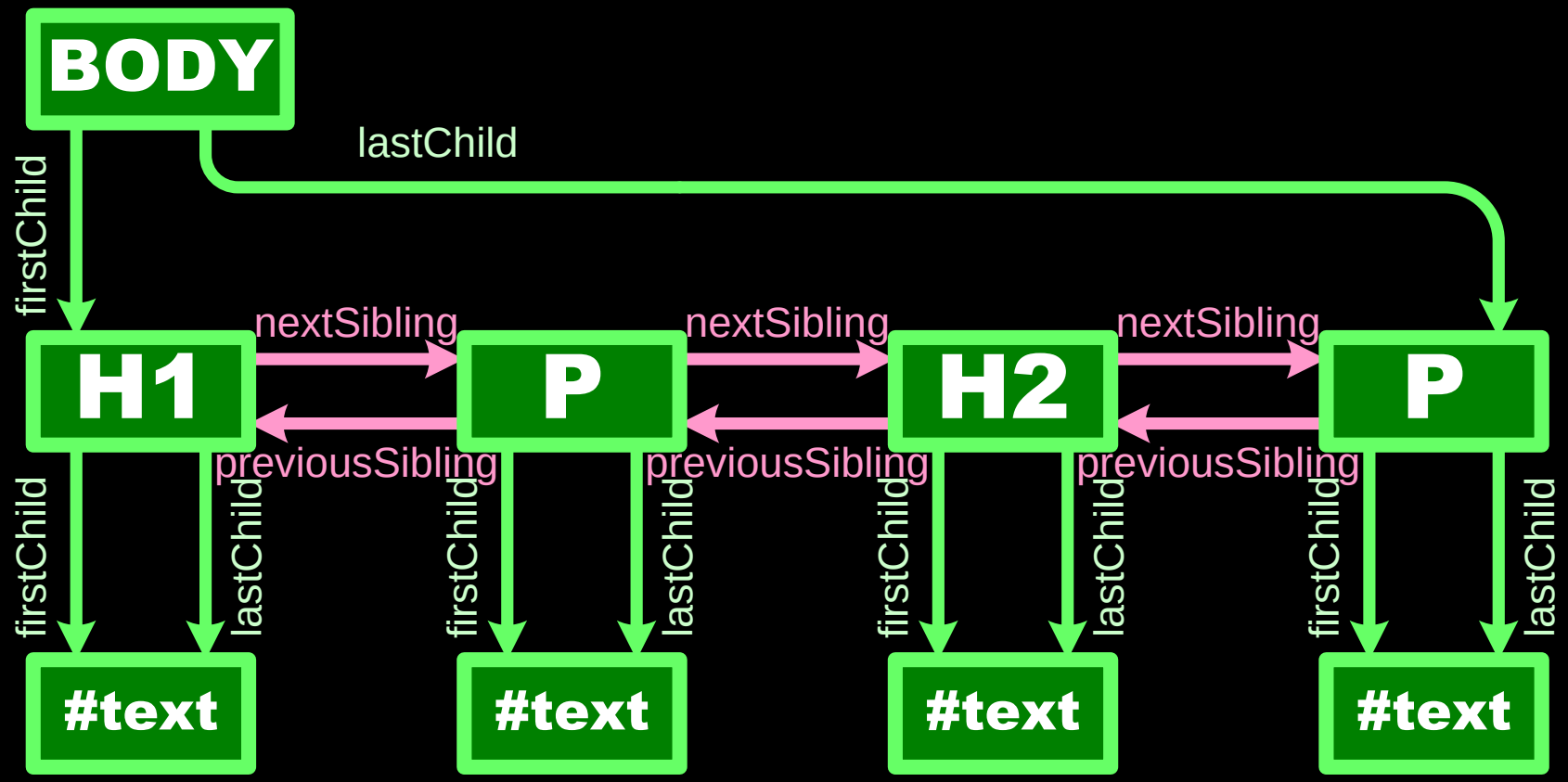



```
<html>
  <body>
    <h1>Heading 1</h1>
        <p>Paragraph.</p>
      <h2>Heading 2</h2>
        <p>Paragraph.</p>
  </body>
</html>
```

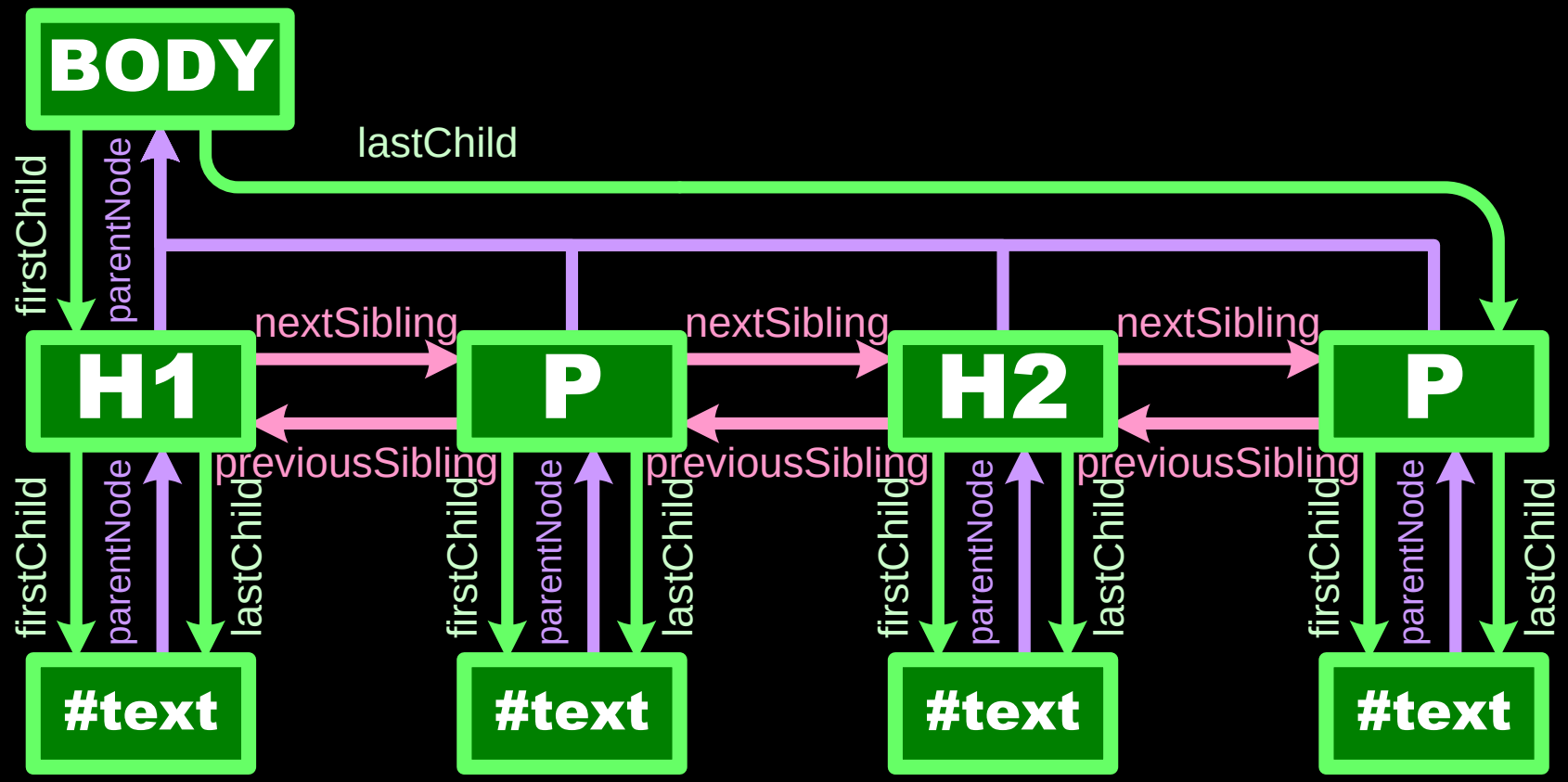
child, sibling, parent



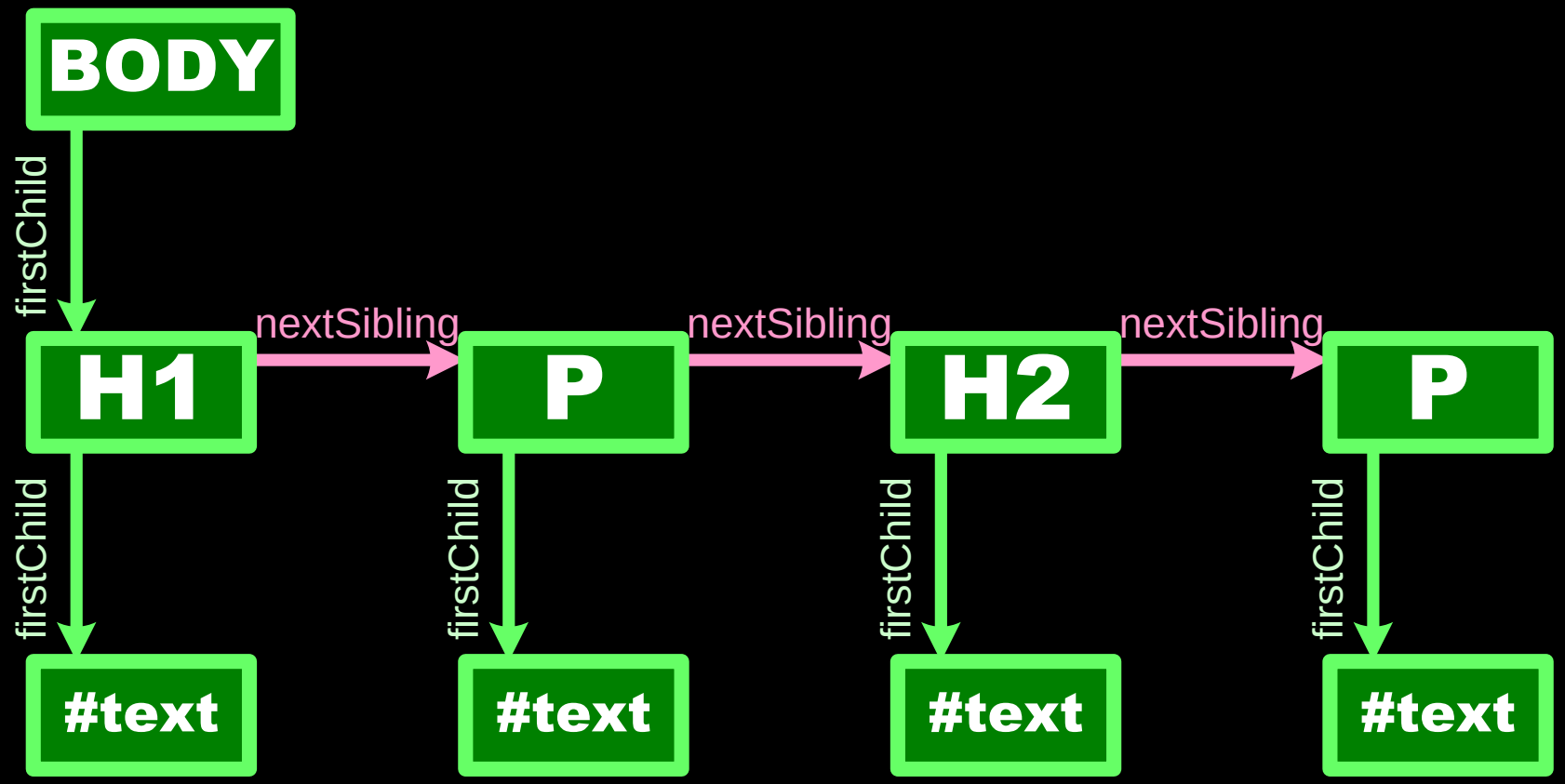
child sibling



child sibling parent



child sibling parent



Simple web page

Chrome File Edit View History Bookmarks Window Help 72% Wed 14:46 Diarmuid O' Connor

dom.html | Diarmuid O'Connor | JavaScript Events Ex | JavaScript HTML DO | dom.html

file://localhost/Users/diarmuidoconnor/Notes/Common2/JavaScript/fundamentalsJS/browserevent...

Heading 1

Paragraph 1

Heading 2

Paragraph 2

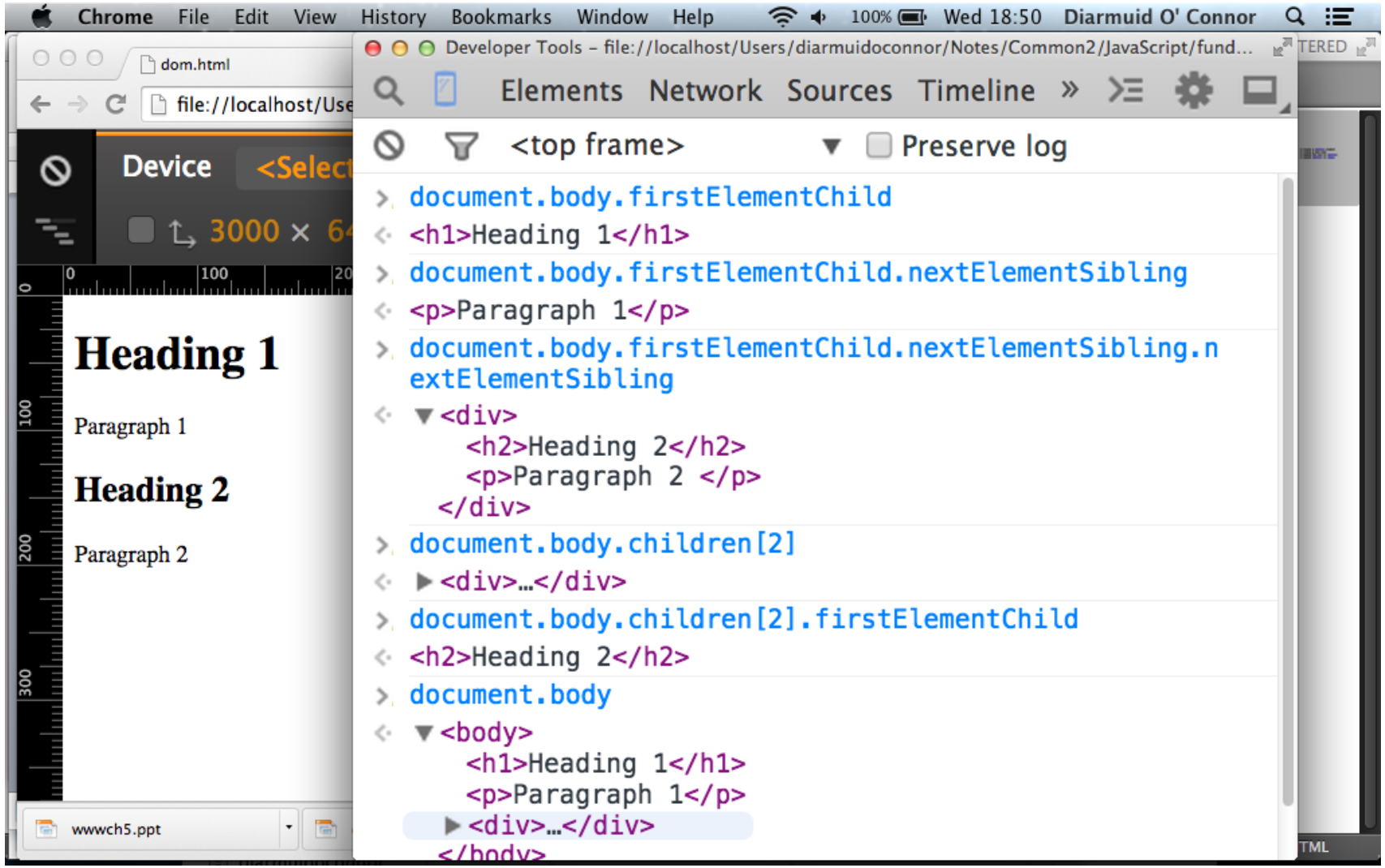
```
1   <!DOCTYPE html>
2   <html>
3   <head>
4
5   </head>
6   <body>
7     <h1>Heading 1</h1>
8     <p>Paragraph 1</p>
9     <div>
10        <h2>Heading 2</h2>
11        <p>Paragraph 2 </p>
12    </div>
13  </body>
```

A child of the body

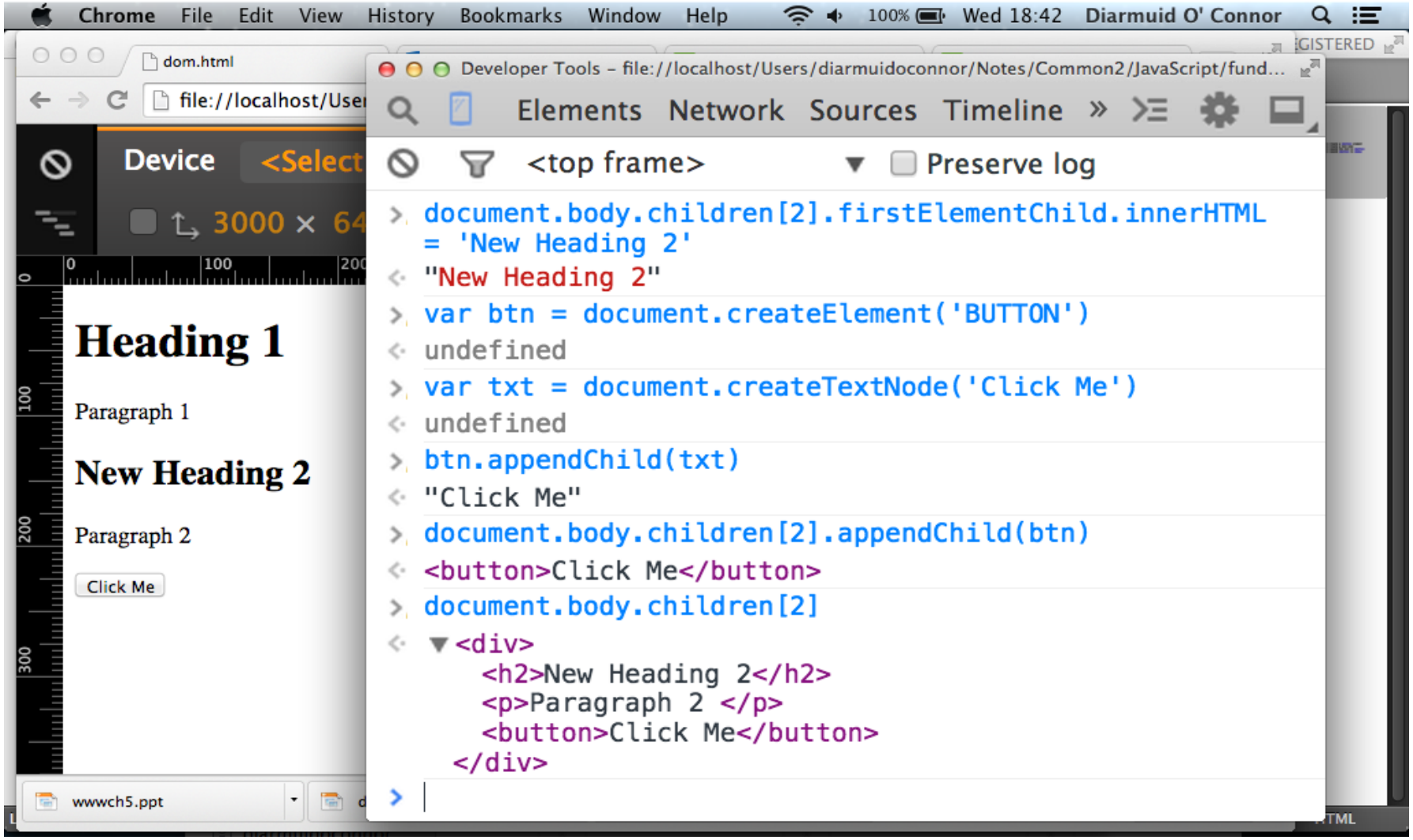
A child of the div

wwwch5.ppt | dom.ppt | hackday.ppt | Show All

Navigating the DOM



Amending the DOM



Events.

The browser has an event-driven, single-threaded, asynchronous programming model.

- **Examples of events**
  - **A mouse click**
  - **A web page or an image loading**
  - **'Mousing' over a hot spot on the web page**
  - **Selecting an input box in an HTML form**
  - **Submitting an HTML form**
  - **A keystroke**

- We can assign event handlers (JS function) to a DOM element.
  - Browser manages handler execution in asynchronuous manner

Event types.

- onabort - Loading of an image is interrupted
- onblur - An element loses focus
- onchange - The content of a field changes
- onclick - Mouse clicks an object
- ondblclick - Mouse double-clicks an object
- onerror - An error occurs when loading a document or an image
- onfocus - An element gets focus
- onkeydown - A keyboard **key is pressed**
- onkeypress - A keyboard key is pressed or held down
- onkeyup - A keyboard key is released
- onload - A page or an image is finished loading
- onmousedown - A mouse button is pressed
- onmousemove - The mouse is moved

Event types.

- onmouseout - The mouse is moved off an element
- onmouseover - The mouse is moved over an element
- onmouseup - A mouse button is releas
- onreset - The reset button is clicked
- onresize - A window or frame is resized
- onselect - Text is selected
- onsubmit - The submit button is clicked
- onunload - The user exits the page

Event Handlers.

- An event handlers/listeners **can be associated with a web page element for specific event types..**
- **Two programming styles:**
  1. Imperative:

     ***dom_node*.addEventListener(*type*, *func*, false)**

  2. Declarative:

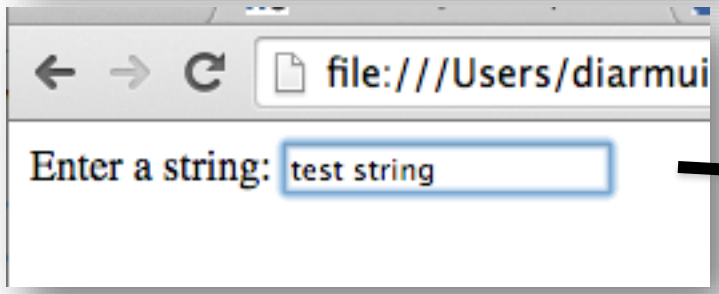
     <tagName on{type}=‘funcName’ …….>

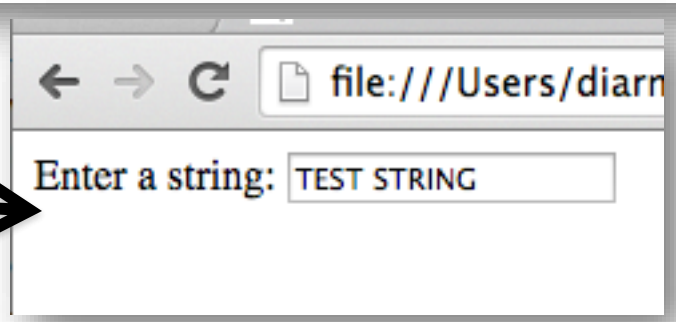
Event Handlers (Declarative style)

Event handler / listener

See 02_1_onchange.html

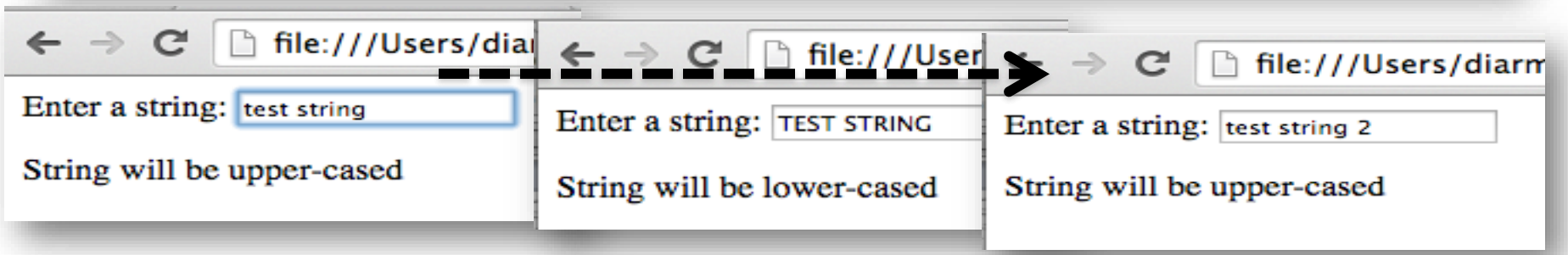
```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <script>
5  function upperCase() {
6      var element = document.getElementById("demo")
7      element.value = element.value.toUpperCase()
8  }
9  </script>
10 </head>
11 <body>
12     <span>Enter a string: <input type="text" id="demo" onchange="upperCase()">
13     </span>
14 </body>
```





- See 02_2_onchange.html
- Imperative style

```html
<html>
<head>
<script>
function upperCase() {
    var element = document.getElementById("demo") ;
    element.value = element.value.toUpperCase();
    // Switch event handler
    element.removeEventListener('change',upperCase );
    element.addEventListener('change',lowerCase , false);
    document.getElementsByTagName('p')[0].innerHTML =
                                'String will be lower-cased on change';
}

function lowerCase(event1) {
  var element =  event1.srcElement   // event has global scope
    element.removeEventListener('change', lowerCase)
    element.addEventListener('change',upperCase , false)
    element.value = element.value.toLowerCase()
    document.getElementsByTagName('p')[0].innerHTML =
                                'String will be upper-cased on change'
}
</script>
</head>
<body>
   <span>Enter a string: <input type="text" id="demo" onchange="upperCase()">
    </span>
<p>String will be upper-cased on change</p>
</body>
```



DOM API → JQuery API.

- The DOM API is not developer-friendly.
- The JQuery JS library (Aug., 2006) improved the developer experience (DX) by:
  - Simplifying event binding and DOM manipulation
  - Providing a common API across multiple browsers
  - Supporting plug-in modules to extend functionality.
- JQuery is built on top of the DOM API.

JQuery.

- JQuery promotes an <u>imperative</u> programming model. (See 03_1_jq-change.html)

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <script src="./jquery.min.js"></script>
5  <script>
6  $(document).ready(function(){
7      $("#demo").change(function(){
8          var newVal = $(this).val().toUpperCase();
9          $(this).val(newVal);
10     })
11 })
12 </script>
13 </head>
14 <body>
15     <span>Enter a string: <input type="text" id="demo"/>
16     </span>
17 </body>
```

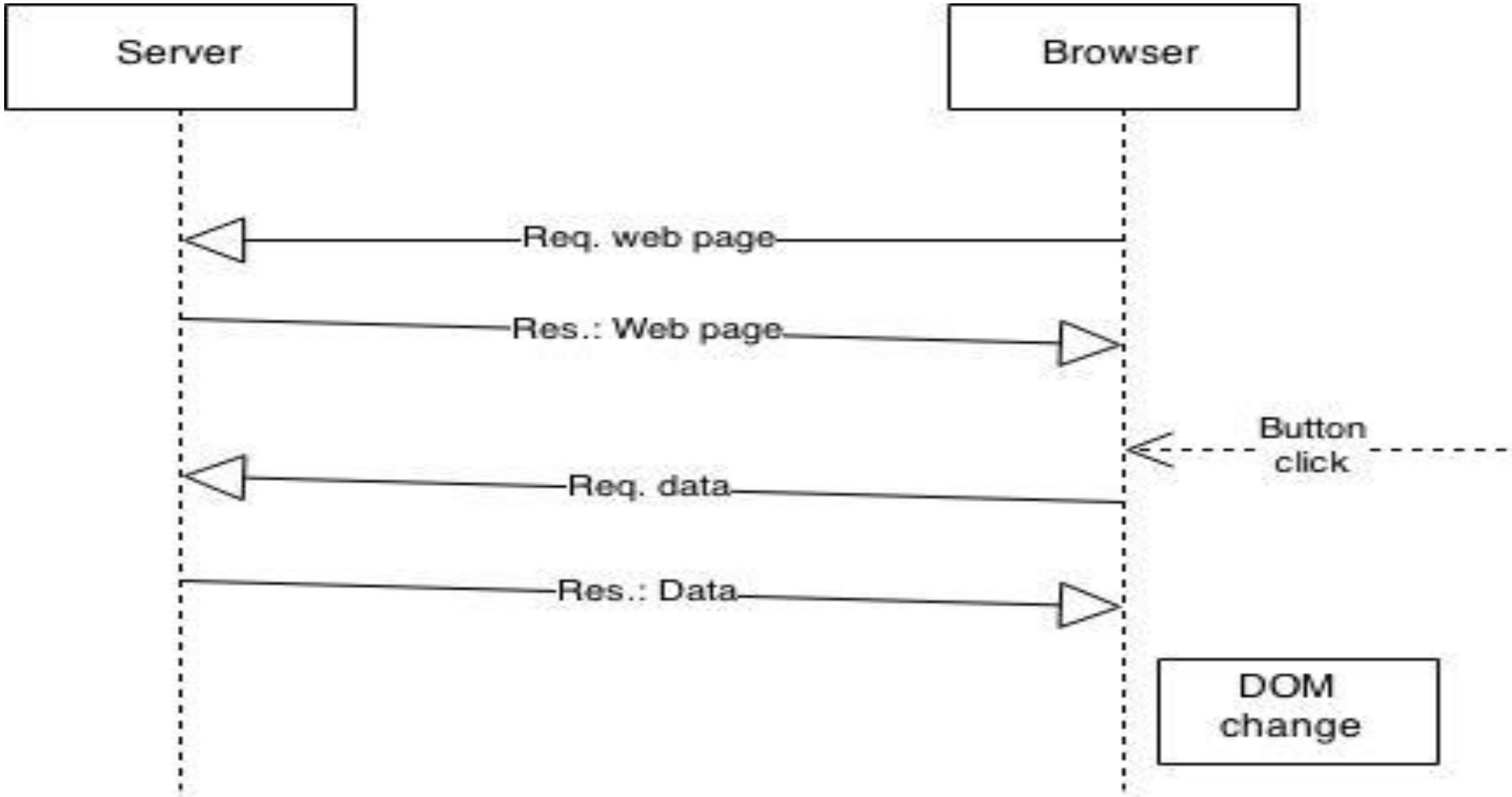
Helpful 'this' binding

JQuery.

- **See also** 03_2_jq-change.html.
  - **Same brhsviour as 02_2_onchange.html but using Jquery.**

Helpful ‘this’ binding

Simple AJAX example (using JQuery) (1/2)



Simple AJAX example (using JQuery) (2/2)

- $.get(URL,callback) **– Send HTTP request to URL; Execute callback function when response arrives.**
- See 04_ajax-jquery.html

Summary

- **The browser stores the ‘current’ web page as a hierarchical network JS objects (nodes)**
- **An API is available to navigate the network.**
  - **DOM API (Default) ; Jquery (Developer-friendly)**
- **The browser provides an event-driven environment.**
- **Event handlers can be linked to nodes for specific events.**
- **Handlers can modify the nodes, causing a re-rendering.**
  - **Result: A web page can be dynamic!!**